WHOOSH

Как мы переезжали с PostgreSQL на Data Lake в AWS

WHOOSH в цифрах

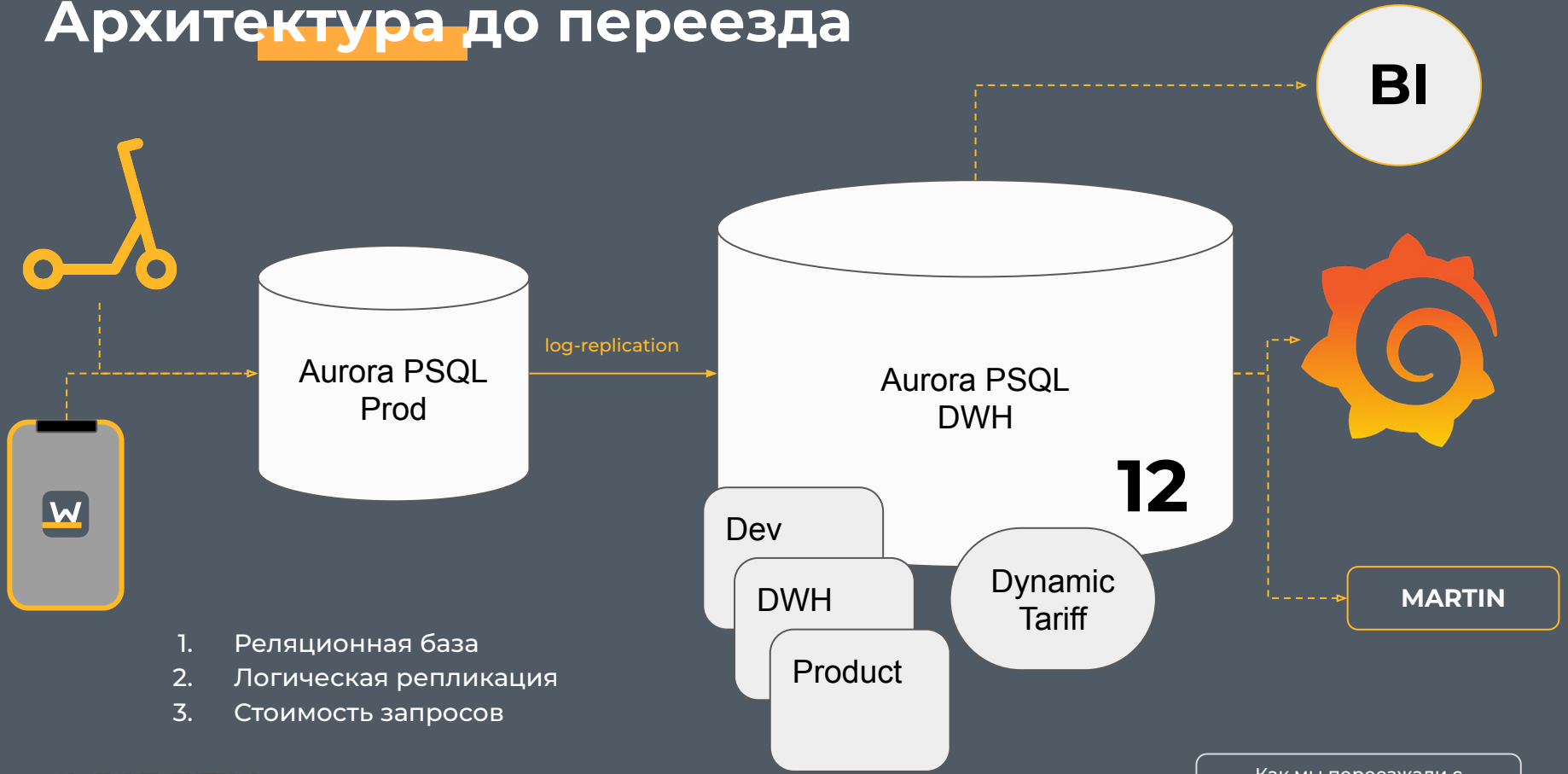
138 000
Самокатов

52
Локации

~30 Тб
Данных

18 600 000
пользователей

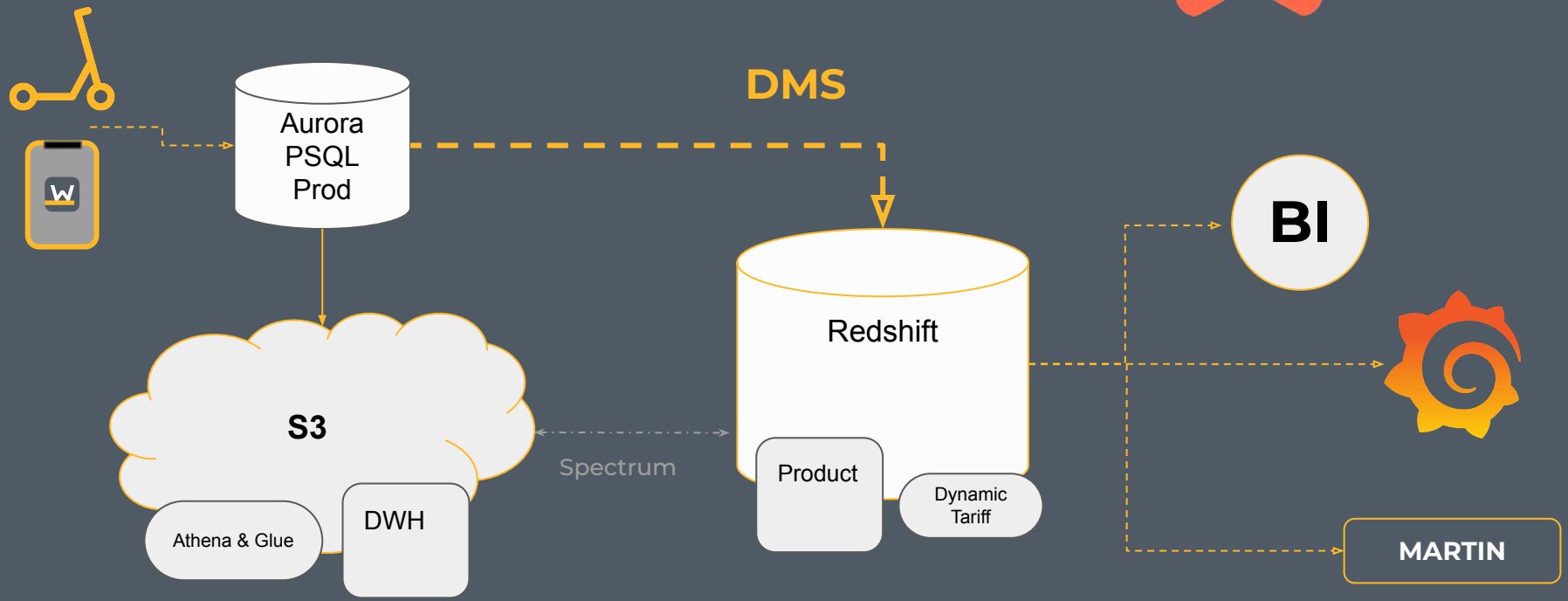
Архитектура до переезда



1. Реляционная база
2. Логическая репликация
3. Стоимость запросов

WHOOSH

Целевая архитектура



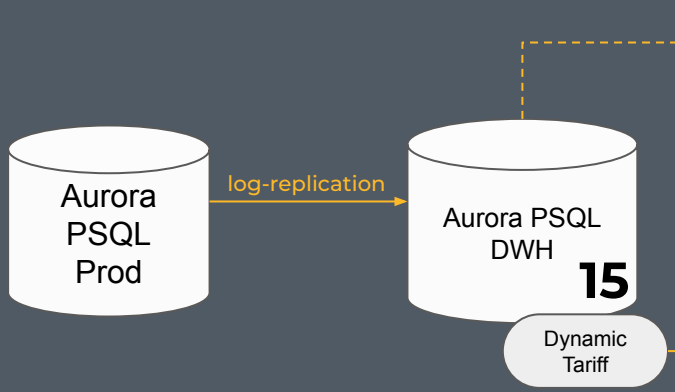
WHOOSH

В Redshift все хорошо, но...

- Отличное решение для DWH
- Подходит для длинных (аналитических) запросов
- Имеет множество привычных функций
- Легко масштабируется

? Функционал для работы с геоданными и почему это важно

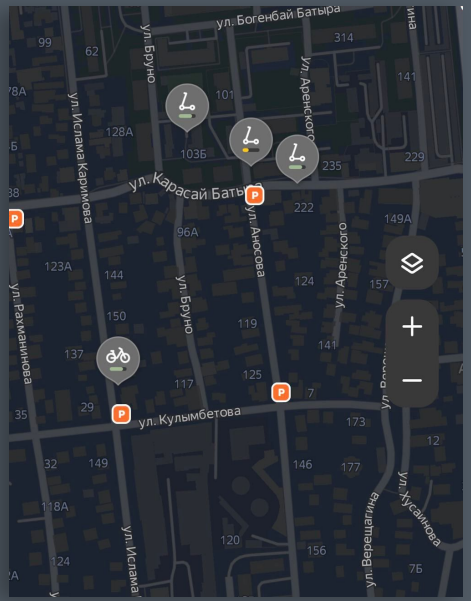
...но PostgreSQL все еще необходим



- Операционная команда
- Аналитическая команда
- Визуализация для внутренних и внешних клиентов

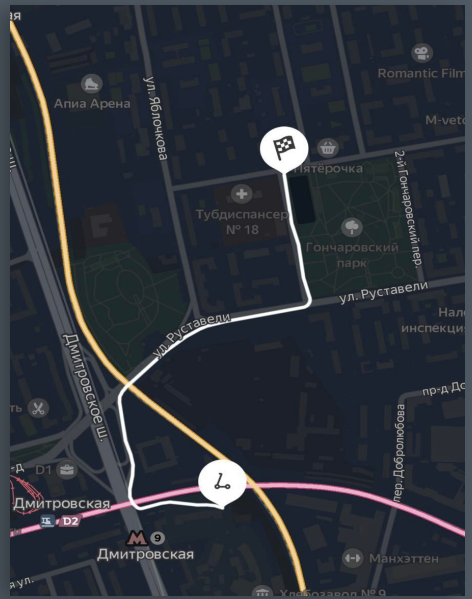
- Сервисная команда
- Бизнес команда
- Множество коротких запросов!

Геоданные



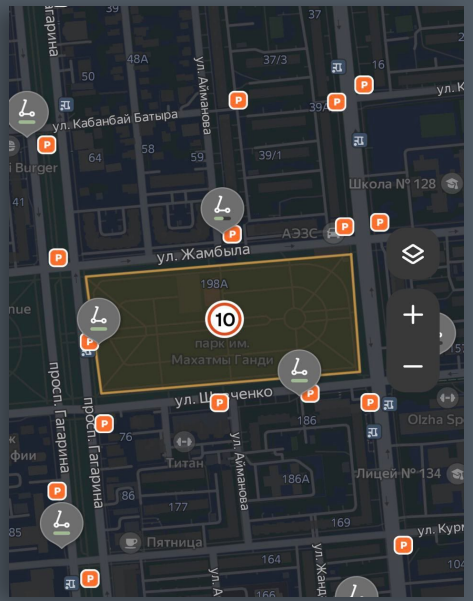
POINT

Парковки / Самокаты



LINESTRING

Треки поездок



POLYGON

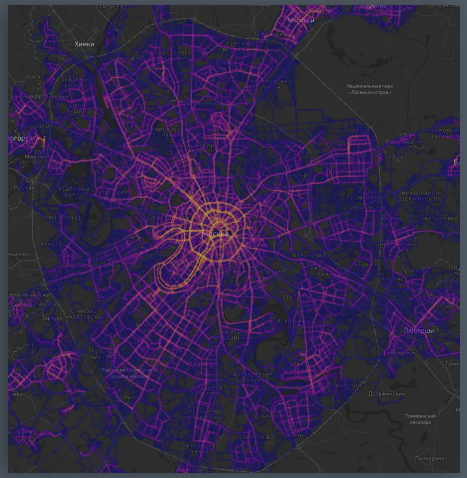
Медленные зоны

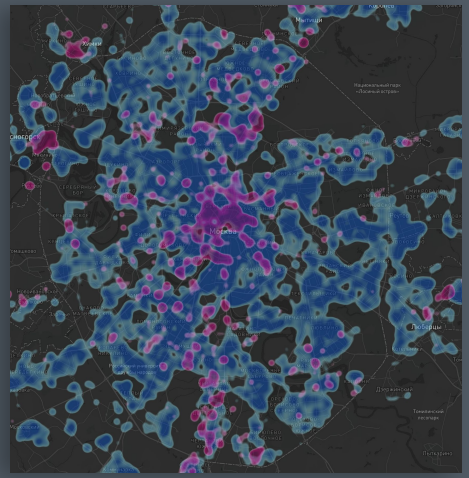
WHOOSH

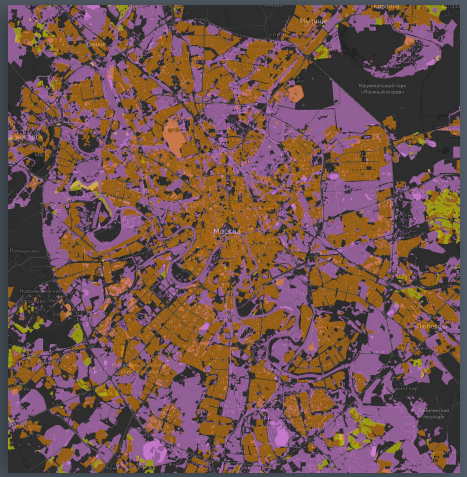
Как мы переезжали с PostgreSQL в DataLake AWS

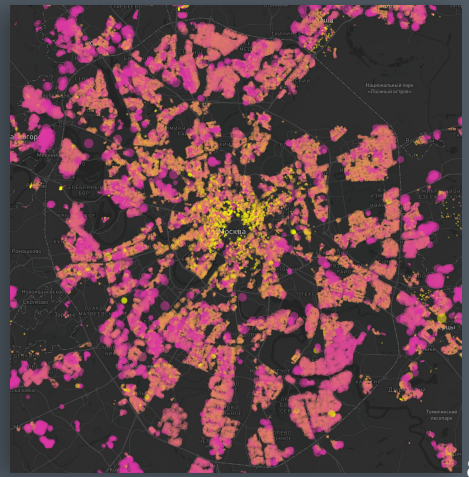
Мартин как продукт

- Визуализация данных в real time для операционной команды
- Слои для аналитической работы
- Open Source инструмент* для создания векторных тайлов с помощью функционала PostGIS
- Сцепка “хранилище + визуализация”









WHOOSH

* Martin - https://github.com/maplibre/martin

AWS Redshift vs PostgreSQL (PostGIS)

Рейтинги	Топ-1 Spatial DBMS[1] Топ-4 в общем рейтинге DBMS	- Топ-35 в общем рейтинге DBMS
Пространственные функции	~367 [2] шт	~107 [3] шт
Разность функций	ST_Makevalid()	Error 🤷‍♂️

SQL Error [XX000]: ERROR: lwgeom_unaryunion_prec: GEOS Error: TopologyException: Input geom 0 is invalid: Self-intersection at or near point 37.322033422436597 55.642111042943831 at 37.322033422436597 55.642111042943831

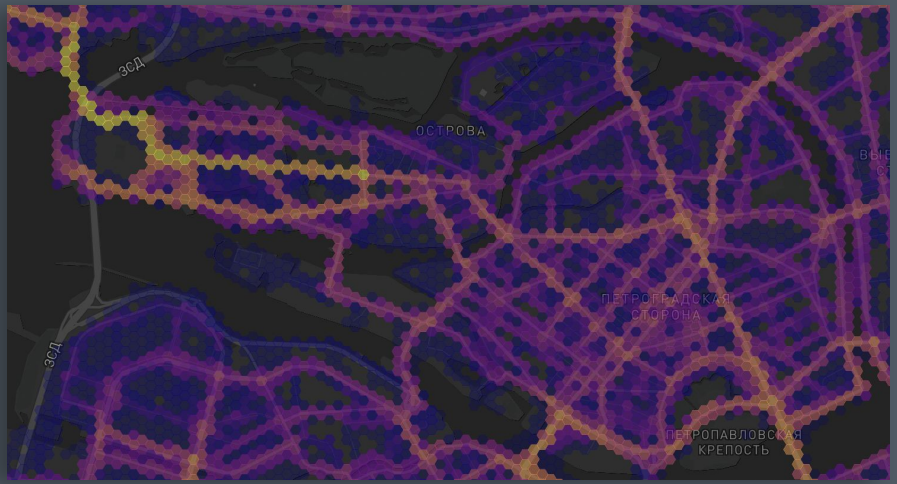
WHOOSH

1. DB Ratings - https://db-engines.com/en/ranking/
2. PostGIS - https://postgis.net/docs/reference.html
3. Redshift Spatial Functions - https://docs.aws.amazon.com/redshift/latest/dg/geospatial-functions.html

Use-cases пространственных функций

Aurora

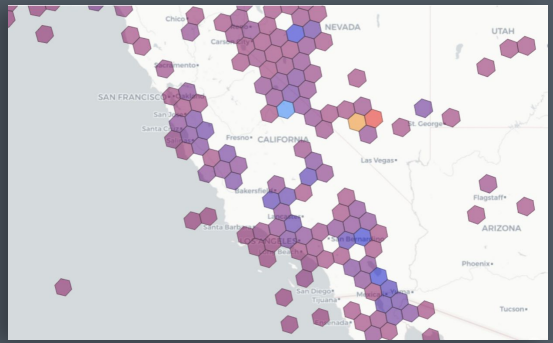
ST_HexagonGrid /
ST_VoronoiPolygons



Redshift

Amazon Athena

UDFs / AWS Lambda /
Ubers H3 [1]

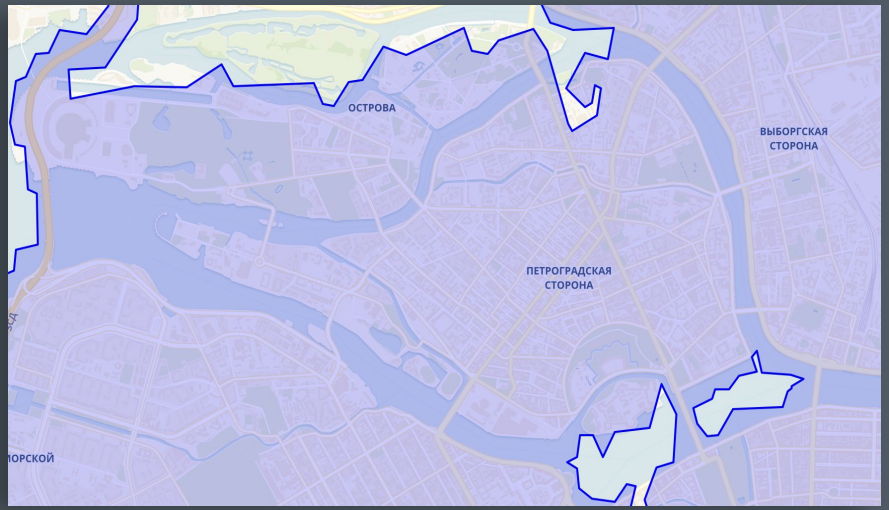


1. Extended geospatial queries -
https://aws.amazon.com/blogs/big-data/extend-geospatial-queries-in-amazon-athena-with-udfs-and-aws-lambda/

Use-cases пространственных функций

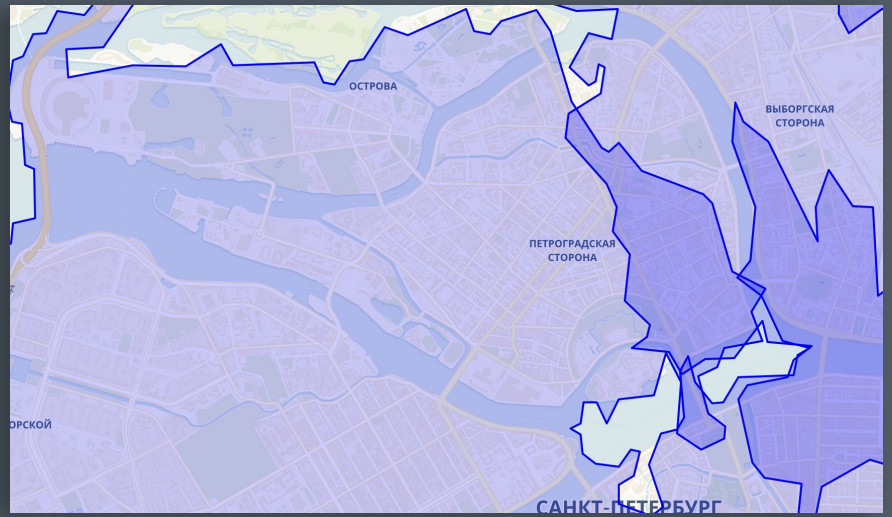
Aurora

ST_Union



Redshift

ST_Collect

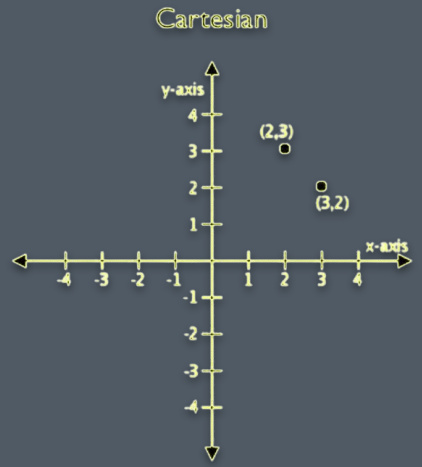


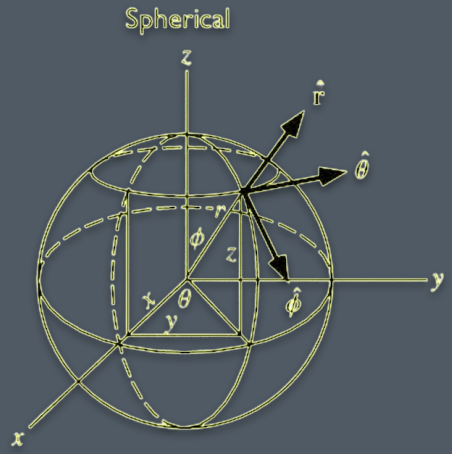
Глоссарий – типы данных в гео

- Geometry/Geography [1]
- WKT – Well-Known Text
- WKB / EWKB -
  (Extended Well-Known Binary)
- Hex-encoded EWKB

6 decimal places – individual humans [2]

‘SRID=4326;
MULTIPOLYGON (((37.290502 55.801989, 37.295422 55.802997, 37.296632 55.803202, 37.297769 55.803354....)))’





1. Geography/Geometry - https://postgis.net/workshops/postgis-intro/geography.html#geography
2. Decimal degrees - https://en.wikipedia.org/wiki/Decimal_degrees

Глоссарий – типы данных в гео

- Geometry/Geography
- WKT – Well-Known Text
- WKB / EWKB -
(Extended Well-Known Binary)
- Hex-encoded EWKB

ST_AsText(geometry)

MULTIPOLYGON (((37.290502 55.801989, 37.295422 55.802997, 37.296632 55.803202, 37.297769 55.803354....)))

Глоссарий – типы данных в гео

- Geometry/Geography
- WKT – Well-Known Text
- WKB / EWKB -
  (Extended Well-Known Binary)
- Hex-encoded EWKB

00000000014000000000000000401000000000000000

- 1-byte integer 00 or 0: big endian
- 4-byte integer 00000001 or 1: POINT (2D)
- 8-byte float 4000000000000000 or 2.0: *x*-coordinate
- 8-byte float 4010000000000000 or 4.0: *y*-coordinate[1]

ST_AsEWKB(geometry)
'0103600000001000000560000006'

1. Well-known representation - https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry#Format_variations

Глоссарий – типы данных в гео

- Geometry/Geography
- WKT – Well-Known Text
- WKB / EWKB -
  (Extended Well-Known Binary)
- Hex-encoded EWKB

ST_AsHEXWKB(geometry)
‘0100060000000100000006ab0bcd71d’

1	1	11	B	21	15
2	2	12	C	22	16
3	3	13	D	23	17
4	4	14	E	24	18
5	5	15	F	25	19
6	6	16	10	26	1A
7	7	17	11	27	1B
8	8	18	12	28	1C
9	9	19	13	29	1D
10	A	20	14	30	1E

Особенности работы с геоданными

Вводные для AWS Redshift и наших данных:

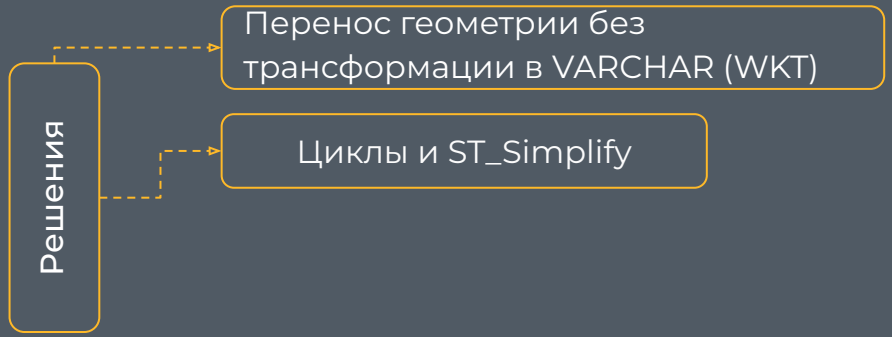
- Ограничение на **1,048,447 bytes** в ячейке[1] для геометрии и **65,535 bytes**[2] для текста (WKT)
- Инструменты для переноса данных (Logical Replication, DMS, Spectrum и т.д)
- ST_MemSize(geometry) -> 10% raw полигональных данных не помещаются в Redshift + аналитика

1. Considerations when using spatial data with Amazon Redshift - https://docs.aws.amazon.com/redshift/latest/dg/spatial-limitations.html
2. Storage and ranges - https://docs.aws.amazon.com/redshift/latest/dg/r_Character_types.html

Особенности работы с геоданными

Вводные для AWS Redshift и наших данных:

- Ограничение на **1,048,447 bytes** в ячейке[1] для геометрии и **65,535 bytes**[2] для текста (WKT)
- Инструменты для переноса данных (Logical Replication, DMS, Spectrum и т.д)
- ST_MemSize(geometry) -> 10% raw полигональных данных не помещаются в Redshift + аналитика

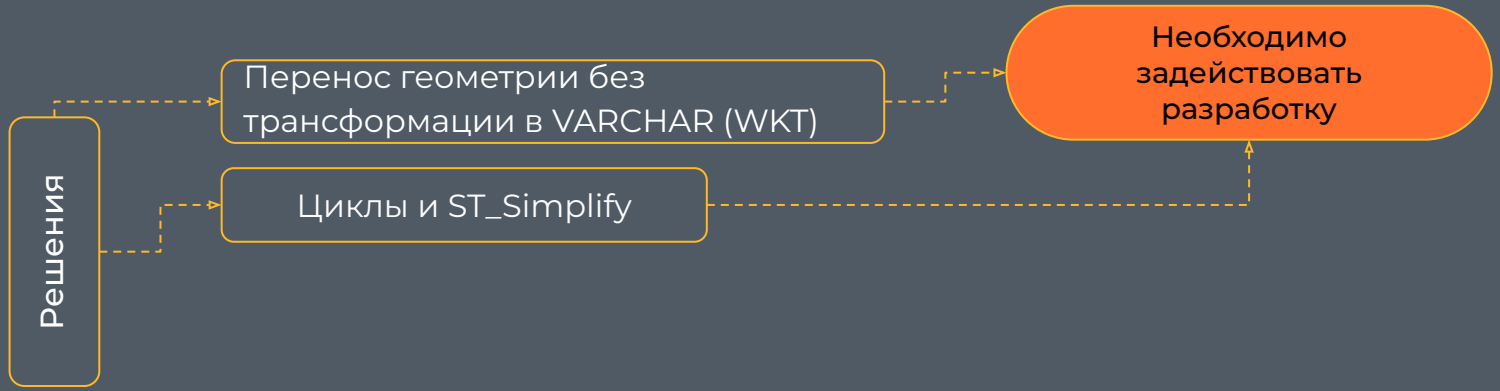


1. Considerations when using spatial data with Amazon Redshift - https://docs.aws.amazon.com/redshift/latest/dg/spatial-limitations.html
2. Storage and ranges - https://docs.aws.amazon.com/redshift/latest/dg/r_Character_types.htm

Особенности работы с геоданными

Вводные для AWS Redshift и наших данных:

- Ограничение на **1,048,447 bytes** в ячейке[1] для геометрии и **65,535 bytes**[2] для текста (WKT)
- Инструменты для переноса данных (Logical Replication, DMS, Spectrum и т.д)
- ST_MemSize(geometry) -> 10% raw полигональных данных не помещаются в Redshift + аналитика

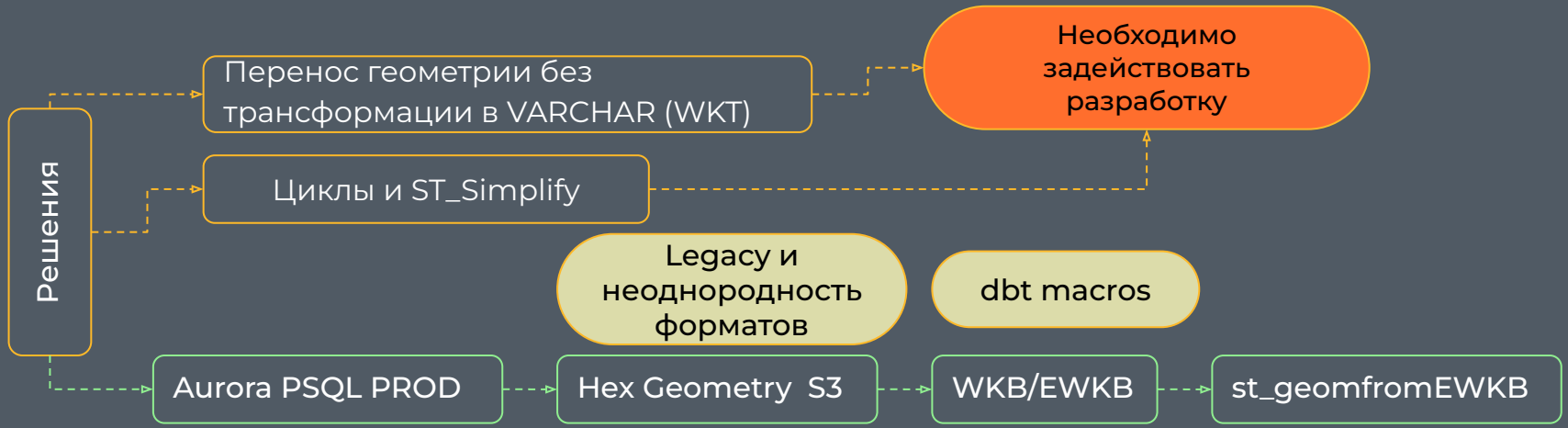


1. Considerations when using spatial data with Amazon Redshift - https://docs.aws.amazon.com/redshift/latest/dg/spatial-limitations.html
2. Storage and ranges - https://docs.aws.amazon.com/redshift/latest/dg/r_Character_types.htm

Особенности работы с геоданными

Вводные для AWS Redshift и наших данных:

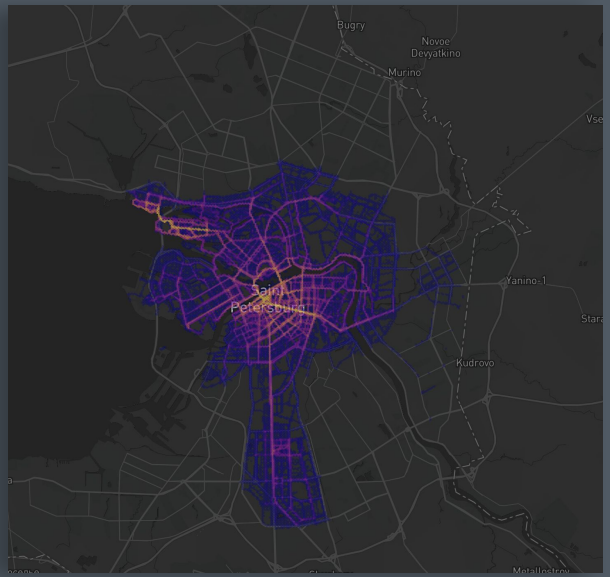
- Ограничение на **1,048,447 bytes** в ячейке[1] для геометрии и **65,535 bytes**[2] для текста (WKT)
- Инструменты для переноса данных (Logical Replication, DMS, Spectrum и т.д)
- ST_MemSize(geometry) -> 10% raw полигональных данных не помещаются в Redshift + аналитика



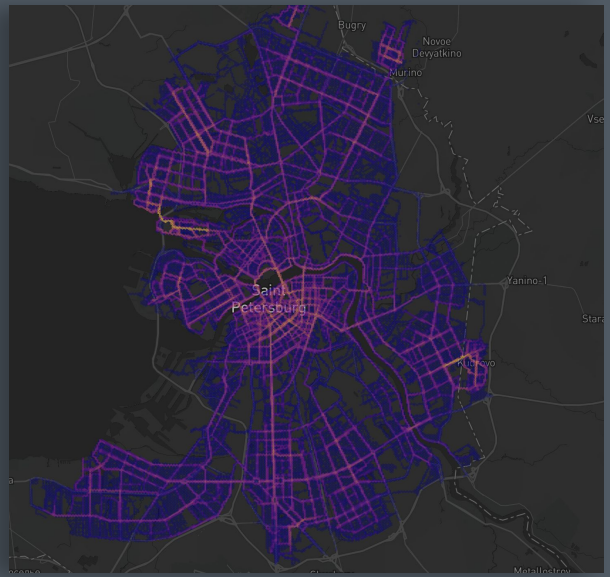
1. Considerations when using spatial data with Amazon Redshift - https://docs.aws.amazon.com/redshift/latest/dg/spatial-limitations.html
2. Storage and ranges - https://docs.aws.amazon.com/redshift/latest/dg/r_Character_types.htm

Большие гео-процессы в Athena

- Рост количества поездок



Поездки 2020



Поездки 2021

Большие гео-процессы в Athena

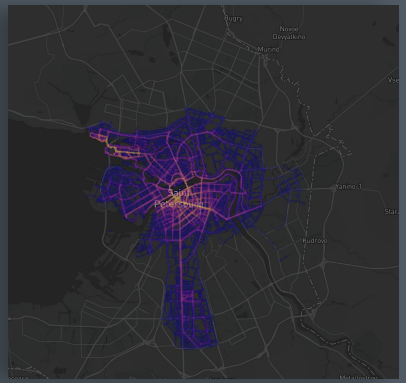
- Партицирование - иерархическая организация на основе метаданных

```yaml
- name: "scooters"
  config:
    materialized: "incremental"
    incremental_strategy: "insert_overwrite"
    external_location: "s3://whoosh/whoosh_stg/scooters_date"
    partitioned_by: [ "date" ]
    format: "PARQUET"
    write_compression: "SNAPPY"
```

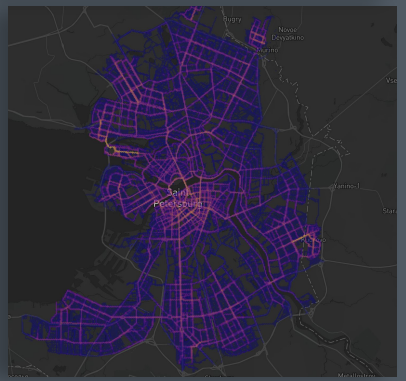
- dbt utils = python + скорость Athena

```
{% set cities = dbt_utils.get_column_values(
    table=ref("city"), column="id"
) %}

{% for city in cities %}
{% if not loop.last %} UNION ALL {% endif %}
{% endfor %}
```



Поездки 2020



Поездки 2021

Как мы победили гео-функции математикой

```
SELECT s.id, p.parking_name
FROM scooters s
CROSS JOIN LATERAL (
    SELECT parking_name
    FROM parkings pa
    WHERE st_dwithin(pa.coordinate = s.coordinate,100)
    order by (pa.coordinate <-> s.coordinate) limit 1) p;
```

Какой парковке принадлежит самокат?

Самокат

Широта	Долгота
55.73397	37.58818

Парковка 1

Широта	Долгота
55.70713	37.69267

Парковка 2

Широта	Долгота
56.12672	40.37445

1. Bulk Nearest Neighbor using Lateral Joins - https://carto.com/blog/lateral-joins

Как мы победили гео-функции математикой

Какой парковке принадлежит самокат?

Самокат

Широта	Долгота
45.07028	39.01160

= round((Широта + Долгота)*1000,0)

Парковка 1

Широта	Долгота
45.0787	39.0009

= round((Широта + Долгота)*1000,0)

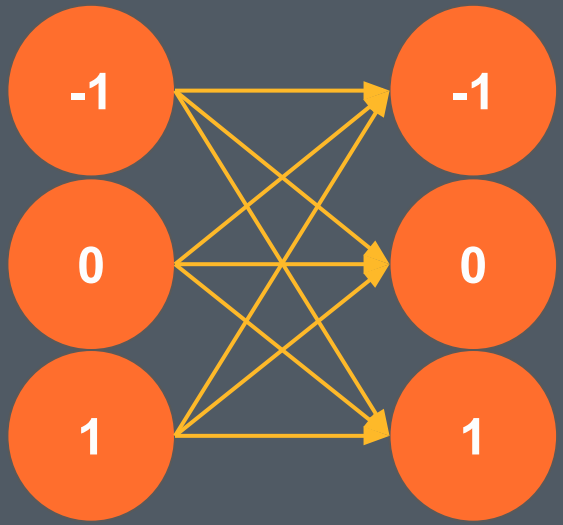
Парковка 2

Широта	Долгота
45.07035	39.01160

= round((Широта + Долгота)*1000,0)

Как мы победили гео-функции математикой

Широта + Долгота → -2, -1, 0, 1, 2 → 5 раз



Возможное округление

```
select s.sam_id
            s.lat,              # Широта и Долгота самоката
            s.lon,
            p.parking_id
            from sams s
        inner join
            parkings as p
            on round(p.lat*1000,0) + round(p.lon*1000,0) =
           round(s.lat*1000,0) + round(s.lon*1000,0) - 2
union all
      …
      = round(s.lat*1000,0) + round(s.lon*1000,0) - 1
```

Скорость выполнения запросов увеличилась **в 10 раз**

Как мы победили гео-функции математикой

Какой парковке принадлежит самокат?

Самокат

Широта	Долгота
45.07028	39.01160

= round((Широта + Долгота)*1000,0)
= 84 082

Парковка 1

Широта	Долгота
45.0787	39.0009

= round((Широта + Долгота)*1000,0)
= 84 080
Фактическое расстояние ~ 1.26 км

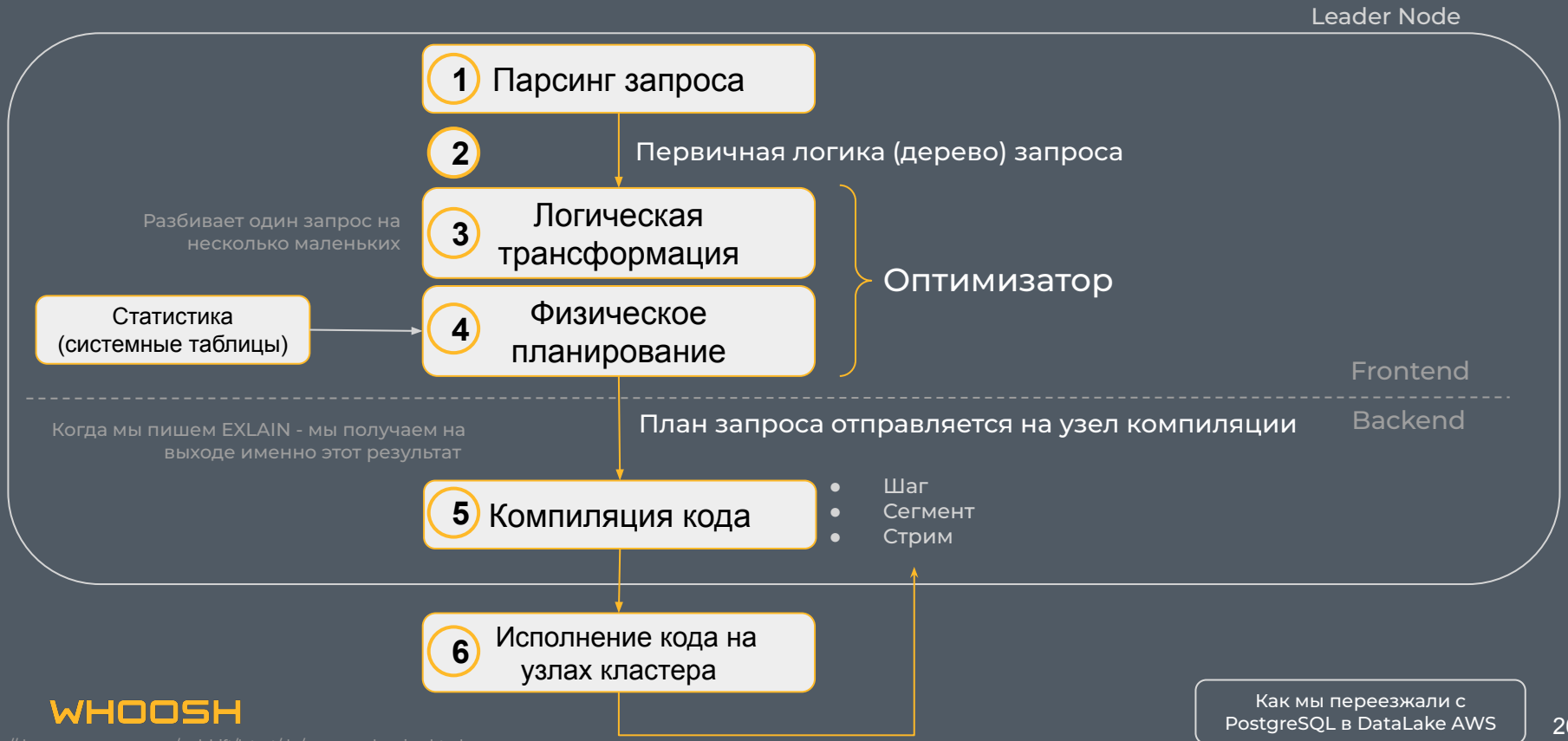
Парковка 2

Широта	Долгота
45.07035	39.01160

= round((Широта + Долгота)*1000,0)
= 84 082
Фактическое расстояние ~7,8 м

WHOOSH

Короткие запросы и их работа в Redshift



https://docs.aws.amazon.com/redshift/latest/dg/c-query-planning.html

Короткие запросы и их работа в Redshift

Возьмем за пример запрос N,
в котором участвуют 2 таблицы
по 150 тыс. строк и 1 left join
Время исполнения ~5,6 с

grouping_flag	region_id	combined_status	battery_power	cnt
TOTAL	NULL	CHANGE_BATTERY	0	102
TOTAL	NULL	CHANGE_BATTERY	1	55
TOTAL	NULL	CHANGE_BATTERY	2	106
TOTAL	NULL	CHANGE_BATTERY	3	263
TOTAL	NULL	CHANGE_BATTERY	4	488

Посмотрим на загрузку:

В сумме требуется скан
> **17 Гб** данных

Result 1 (59)

query	step	rows	bytes	workmem
84535941	2	690176	68773956	4065067008
84536314	0	87	8178	0
84536314	1	4670464	513124782	0
84536314	2	0	0	0
84536314	3	0	0	0
84536314	4	0	0	0
84536314	5	4670464	622826432	0
84536314	0	13915136	1721411444	0
84536314	1	0	0	0
84536314	2	1888256	369199212	4643487744
84536314	2	12026880	1463533256	663355392
84536500	0	1	644	0
84536500	1	0	0	0
84536500	2	0	0	0
84536500	3	0	0	0
84536503	0	0	0	0
84536503	1	0	0	0
84536503	2	0	0	169082880
84536517	0	0	0	0
84536517	1	0	0	0

Посмотреть на загрузку можно по полю workmem

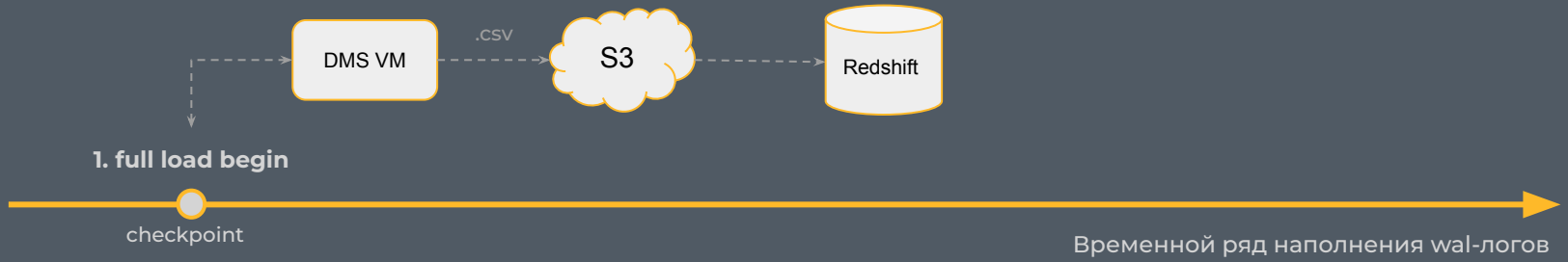
Data Migration System

- Инструмент, который нам сильно облегчил жизнь по переносу таблиц в режиме “онлайн + CDC”
- Не затрагивает мощности исходной базы (за исключением времени, когда требуется Full Load при включенном CDC)
- Простой пользовательский путь

Временной ряд наполнения wal-логов

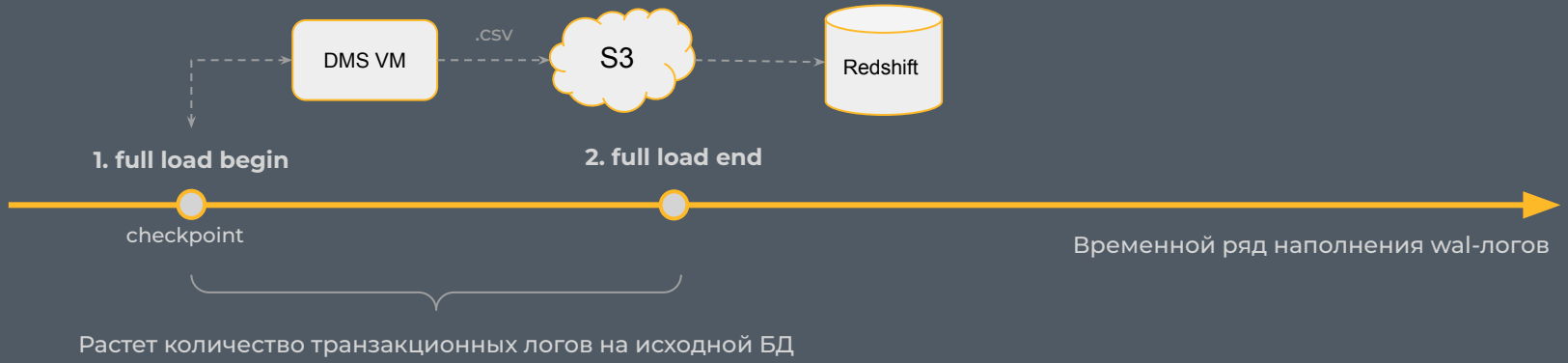
Data Migration System

- Инструмент, который нам сильно облегчил жизнь по переносу таблиц в режиме “онлайн + CDC”
- Не затрагивает мощности исходной базы (за исключением времени, когда требуется Full Load при включенном CDC)
- Простой пользовательский путь



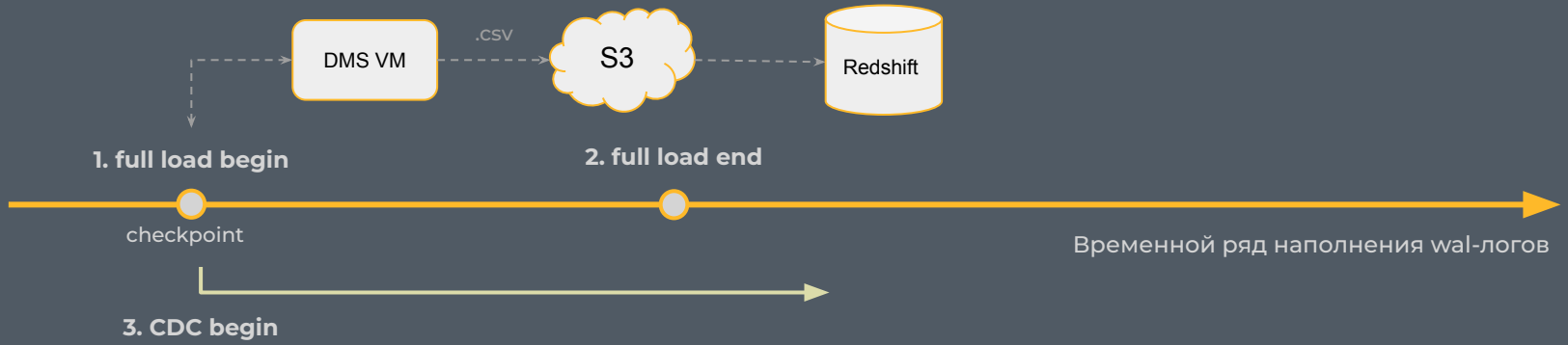
Data Migration System

- Инструмент, который нам сильно облегчил жизнь по переносу таблиц в режиме “онлайн + CDC”
- Не затрагивает мощности исходной базы (за исключением времени, когда требуется Full Load при включенном CDC)
- Простой пользовательский путь



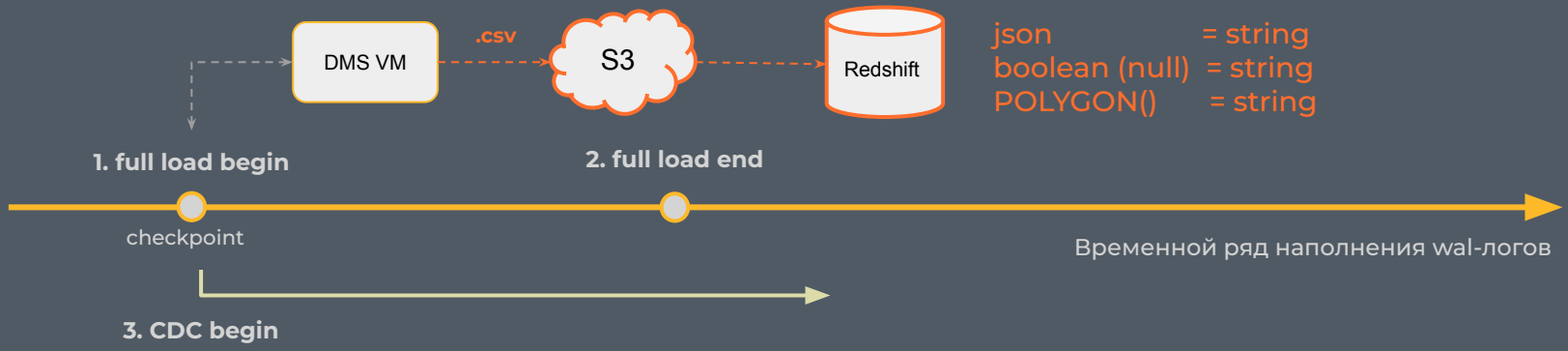
Data Migration System

- Инструмент, который нам сильно облегчил жизнь по переносу таблиц в режиме “онлайн + CDC”
- Не затрагивает мощности исходной базы (за исключением времени, когда требуется Full Load при включенном CDC)
- Простой пользовательский путь



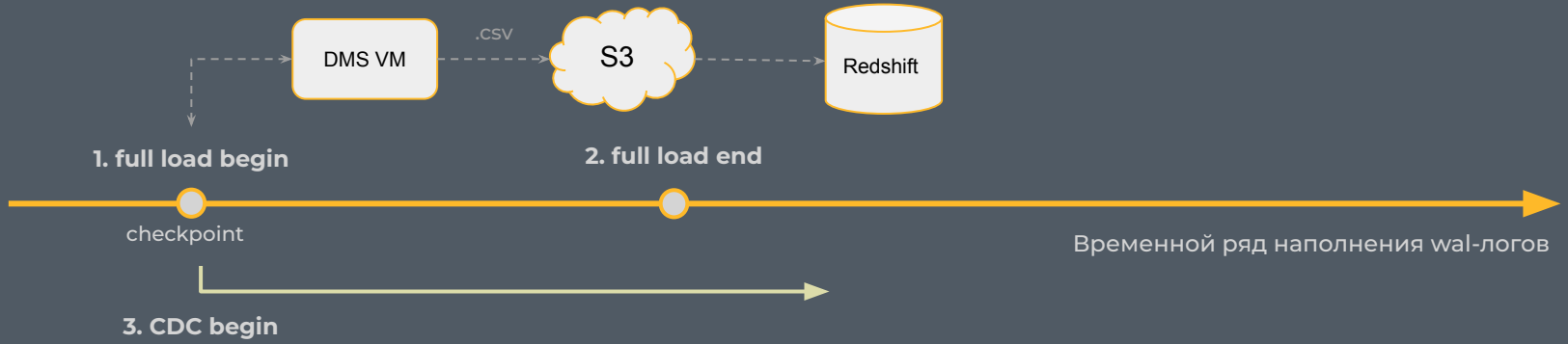
Data Migration System

- Врет в типах данных
- Не поддерживает транзакции DDL, которые были сделаны внутри procedure
- С последним обновлением, сломан параллелизм при заполнении данных в Редшифт
- Ограниченные функции по отладке ошибок



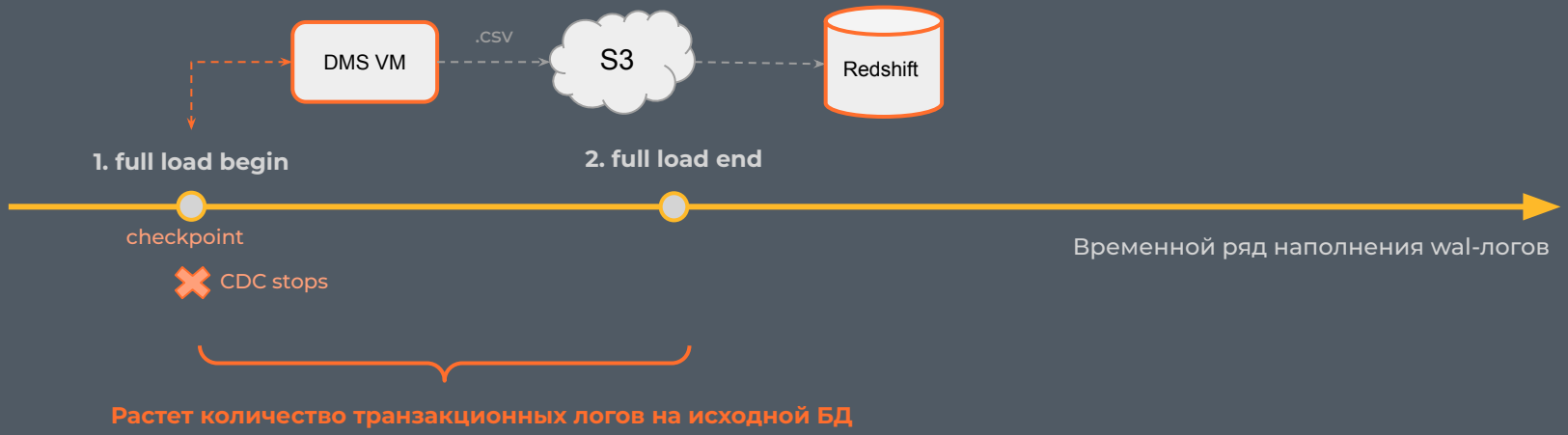
Data Migration System

- Врет в типах данных
- Не поддерживает транзакции DDL, которые были сделаны внутри procedure
- С последним обновлением, сломан параллелизм при заполнении данных в Редшифт
- Ограниченные функции по отладке ошибок

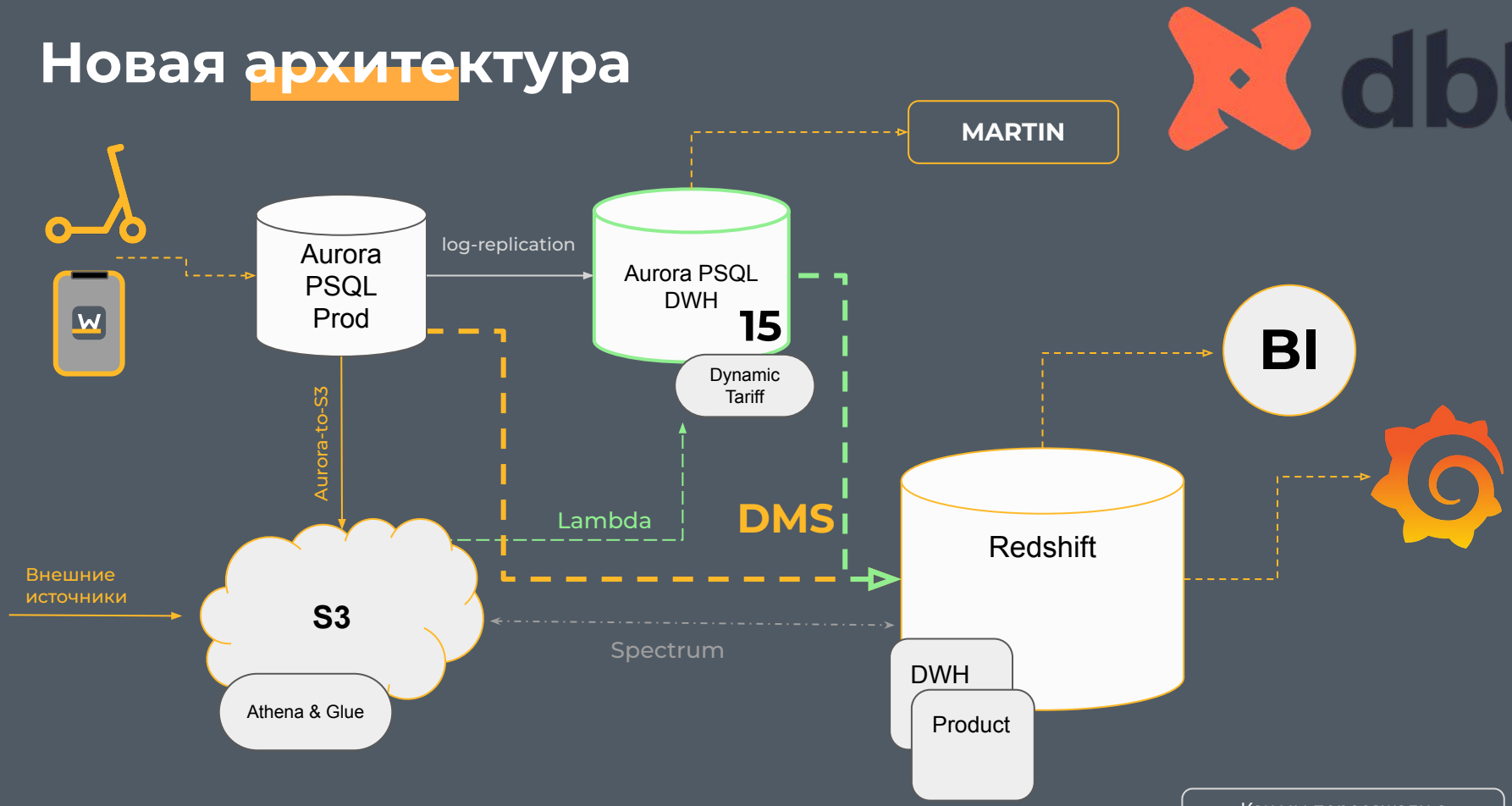


Data Migration System

- Врет в типах данных
- Не поддерживает транзакции DDL, которые были сделаны внутри procedure
- С последним обновлением, сломан параллелизм при заполнении данных в Редшифт
- Ограниченные функции по отладке ошибок



Новая архитектура

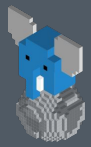


Что мы приобрели?

1. Выстроенные процессы data-pipeline
2. Увеличенную скорость работы всей отчетности и доставки инсайдов до бизнеса
3. Существенно оптимизированные затраты на хранилище
4. Рост в компетенциях работы с AWS
5. Гибкое и легко масштабируемое архитектурное решение

Куда дальше?

elementary



1. DataQuality = Data Observability + Data Anomaly (Unsupervised Machine Learning)
2. PostGIS - это хорошо, но может Python Lambda?
3. Geoparquet[1]?
4. LakeHouse?
5. Analytics Engineers

1. https://medium.com/@dr.jiayu/harnessing-overture-maps-data-apache-sedonas-journey-from-parquet-to-geoparquet-d99f7767a499

Лучше один раз сделать, чем много раз спорить

WHOOSH

Водите аккуратно, с ветерком

Спасибо за внимание